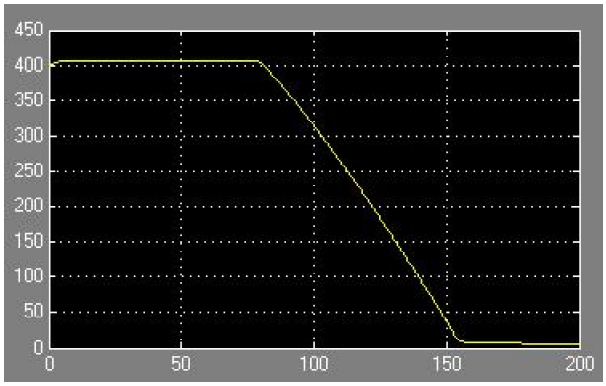
Written by: Chieh Wu

April / 19 /2006

Designing an Automatic Carrier Landing System

This paper will show you a step by step approach to design a basic controller. More specifically, we are designing a controller for an aircraft to land on a carrier. The aircraft starts off at an altitude of 400 meters and it will land on the carrier after sometime. The graphic below is shows the expected trajectory of height versus time.



For this paper, we are going to assume that the aircraft is already aligned to the carrier so we only need to deal with the height and the x position. The idea of controller design will not change due to this simplification. First of all, the aircraft's longitudinal equation of motion is given by these following equations.

$$\dot{\alpha} = \frac{g}{V}(n_z + \cos\gamma_v) - \frac{L}{mV}$$

$$\dot{V} = \frac{T - D}{m} - g\sin\gamma_v$$

$$\dot{\gamma_v} = \frac{L - mg\cos\gamma_v}{mV}$$

$$\dot{x} = V\cos\gamma_v$$

$$\dot{h} = V\sin\gamma_v$$

$$L = \frac{1}{2}\rho(V + V_w)^2 S(C_{L0} + C_{L\alpha}(\alpha + \alpha_w))$$

$$D = \frac{1}{2}\rho(V + V_w)^2 S(C_{D0} + C_{D\alpha}(\alpha + \alpha_w))$$

states of the aircraft

α = angle of attack

V = air speed

γ_v = flight path angle

x = inertial x position

h = altitude

L = lift

D = drag

input into the aircraft

n_z = normal acceleration

T = Thrust

Constants for the aircraft (in SI)

ρ = air density = 1.225

S = wing area = 40

$C_{L0}, C_{L\alpha}, C_{D0}, C_{D\alpha}$ = parameters for the wing area.

C_{L0} = 0.3

C_{D0} = 0.02

$C_{L\alpha}$ = 5.92

$C_{D\alpha}$ = 0.15

V_ω , α_w = process noises representing the effect of wind and g is the acceleration due to gravity.

m = mass = 10,000

g = gravity = 9.8

The aircraft is equipped with many sensors and it is reasonable to assume that the measurements will be noisy.

$$y = \begin{pmatrix} \alpha \\ V \\ \gamma_v \\ x \\ h \end{pmatrix} + v_{\text{ac}}$$

where v_{ac} is white noise with covariance:

$$v_{\text{ac}} = \begin{pmatrix} (\frac{\pi}{180})^2 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & (\frac{\pi}{180})^2 & 0 & 0 \\ 0 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{pmatrix}$$

The operating points are

$\alpha = \frac{\pi}{36}$, $V = 70$, $\gamma = 0$, x = 10,000, h = 400 , $n_z = 0$, T = 4000

Although these points are not exactly the equalibrium points, but since they are close enough, for the sake of our problem, we will use these points as the equalibrium.

Design approach

The general approach procedure.

1. Build the S-function of the aircraft for the non-linear system and simulate it
2. Linearize the plant
3. Check if the linearized plant is AS about 0 using Lyapunov's indirect method.
4. If the linearized plant is AS, the non-linear plant is Locally AS.
5. Write out the generalized plant
6. Depending on the plant, decide if a Kalman filter is needed
7. Design a LQR to get the constant gain for feedback.

Building the S-function in Matlab

To make the S-function, it is necessary to first write out the data fields. Start a new m file, you can call it anything, but I am going to call it systemdata.m . Write these commands in to the file.

```
%.....................................................................
clear
clc

x0 = [pi/36;70;0;10000;400];
sysdata = struct('nx',5,'nu',2,'ny',5);
sysdata = setfield(sysdata, 'x0', x0);
%.....................................................................
```

Sysdata in this case sets up a structure with different variables.

nx is the number of states, we have 5 $(\alpha, V, \gamma, x, h)$

nu is the number of inputs, we have 2 (n_z, T)

Next, we'll have to write out the S-function that simulates the system. We'll save this function and call it systemfunction.m

```
%.....................................................................
function [sys,x0,str,ts] = Feed(t,x,u,flag,sysdata)
switch flag,
case 0, % set initial state,state space dimensions,etc
sizes = simsizes;
sizes.NumContStates = sysdata.nx;          % number of continuous states
sizes.NumDiscStates = 0;                   % number of discrete states
sizes.NumOutputs = sysdata.ny;             % number of outputs
sizes.NumInputs = sysdata.nu;              % number of inputs
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = sysdata.x0; % initial state
str = [ ];
ts = [0 0];

case 1,                                    % calculate state derivatives
sys = f(x,u,sysdata);                      % function where xdot is calculated
```

```
case 3,                                  % calculate outputs
sys = x;
case {2,4,9}, % remaining cases
sys = [ ];
otherwise error(['Unhandled flag = ',num2str(flag)]);
end

function [xdot] = f(x,u,sysdata)
S = 40; m = 10000; p = 1.225; g = 9.8; CL0 = 0.3; CD0 = 0.02; CLa = 5.92; CDa = 0.15;

xdot = [ g*(u(1) + cos(x(3)))/x(2) - (0.5*p*x(2)2*S*(CL0+CLa*x(1)))/(m*x(2));
         (u(2)-(0.5*p*x(2)2*S*(CD0+CDa*x(1))))/m - g*sin(x(3));
         ((0.5*p*x(2)2*S*(CL0+CLa*x(1)))-m*g*cos(x(3)))/(m*x(2));
         x(2)*cos(x(3));
          x(2)*sin(x(3))];

%.....................................................................
```

This is a standard template for the S function where

case 0

we set up all the initial conditions, such as the number of states, inputs, and outputs. They are all stored in sysdata from the systemdata.m file we just wrote.

case 1

Here, we defined the function we call to define the state equation. The function we call f is defined at the end.

case 3

Here, we define the output. In our case, since the output is the same as the internal state, we simply have an identity matrix.

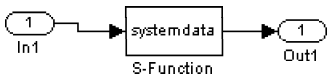
The remaining cases are used for error handling which the details are outside the scope of this paper.

In the f function, we defined the actual state equations. The x(1)...x(5) are the different states $(\alpha, V, \gamma, x, h)$ while u(1) and u(2) are the different inputs (n_z, T).

Once you have finished doing this, fire up simulink by typing simulink in matlab. Go to file/new/model to start a new simulink file. You should see a white window where we can put all the different blocks.

In the Simulink Library Browser the pops up when you type simulink, look for "user-defined functions" under simulink. If you click on that, you will see the S function block showing up on the right side of the window. You want to left click on the S function block and drag it into the white space. Double click on the block and fill in the S function name as well as the S function parameter. The name should be the name of our function file which we called systemfunction. The parameter should be the system data structure we are actually using which we called sysdata. Now save this diagram into the same directory as the systemdata.m. Call it systemblocks.

Next go to sources and look for a little bubble called In1, also drag this bubble into the white space. Do the same thing for sink and drag the out1 into the white sapce. The In1 is the input for our block while out1 is the output. You can connect the block by left clicking on the tip of the blocks and drag a line out for connection. You should end up with something like this.

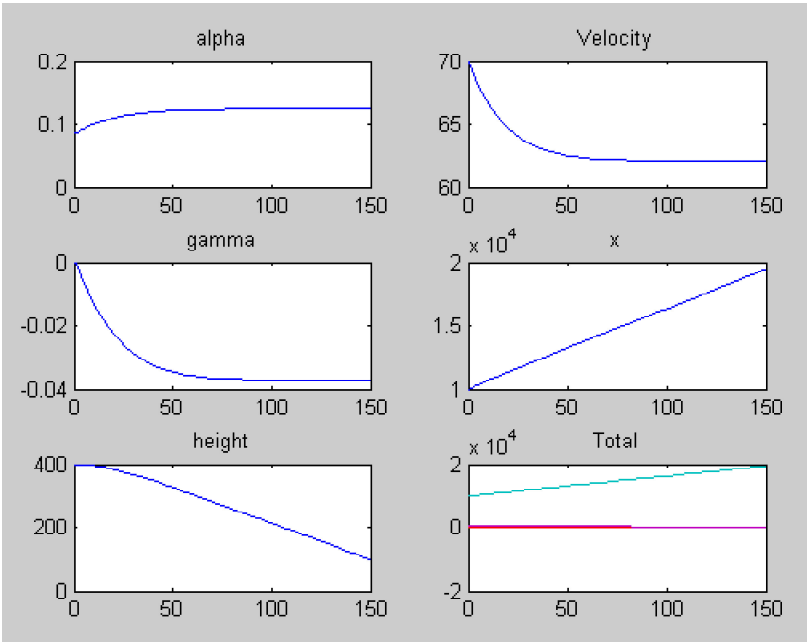


All the block that we will use will be inside the simulink section.

To simulate this block, go back to the systemdata.m and put these commands at the end.

```
time = 150;
[t,x,y] = sim('systemblocks',time);
subplot(3,2,1);plot(t,x(:,1));title('alpha');
subplot(3,2,2);plot(t,x(:,2));title('Velocity');
subplot(3,2,3);plot(t,x(:,3));title('gamma');
subplot(3,2,4);plot(t,x(:,4));title('x');
subplot(3,2,5);plot(t,x(:,5));title('height');
subplot(3,2,6);plot(t,y);title('Total');
```

Once you have finished this step, you are ready to run the systemdata.m. You should see something like this as the output.



Now that we have simulated the plant, we are ready to design the actual controller by linearizing the system.

Linearization of the plant

The system of equations that describes the aircraft are non-linear equations. A general approach towards these types of equations is to linearize them.

According to the Lyapunov's indirect method.

1. If the linearization of a non-linear system is Asymptotically stable about 0, then the nonlinear system is locally Asympototic about x_{eq}.

2. If the linearization of a non-linear system is unstable about 0, then the non-linear system is also unstable.

Using this principle, if the linearized version of the plant is AS, we can conclude that the non-linearized plant is locally AS around the operating points.

To linearize the plant, we first need to find the Jacobian of the non-linear system. The equation for the Jacobian of a system:

$$\frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdot & \cdot & \cdot & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdot & \cdot & \cdot & \frac{\partial f_2}{\partial x_n} \\ \cdot & & & & & \\ \frac{\partial f_n}{\partial x_1} & \cdot & \cdot\cdot & \cdot & \cdot & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

This looks like a lot of work to take the derivative for each equation and each variable. This is why it is normally done in matlab. For our sake, we will do it by hand. The partial derivative for each element of the matrix are the following.

$$\frac{\partial \dot{\alpha}}{\partial \alpha} = \frac{-\rho V^2 S C_{L\alpha}}{2mV} \qquad \frac{\partial \dot{V}}{\partial \alpha} = \frac{-\rho V^2 S C_{D\alpha}}{2m}$$

$$\frac{\partial \dot{\alpha}}{\partial V} = \frac{L - gm(n_z + \cos\gamma)}{V^2 m} \qquad \frac{\partial \dot{V}}{\partial V} = \frac{-\rho V S (C_{D0} + C_{D\alpha}\alpha)}{m}$$

$$\frac{\partial \dot{\alpha}}{\partial \gamma} = \frac{-g\sin\gamma}{V} \qquad \frac{\partial \dot{V}}{\partial \gamma} = -g\cos\gamma$$

$$\frac{\partial \dot{\alpha}}{\partial x} = 0 \qquad \frac{\partial \dot{V}}{\partial x} = 0$$

$$\frac{\partial \dot{\alpha}}{\partial h} = 0 \qquad \frac{\partial \dot{V}}{\partial h} = 0$$

$$\frac{\partial \dot{\alpha}}{\partial n_z} = \frac{g}{V} \qquad \frac{\partial \dot{V}}{\partial n_z} = 0$$

$$\frac{\partial \dot{\alpha}}{\partial T} = 0 \qquad \frac{\partial \dot{V}}{\partial T} = 0$$

$$\frac{\partial \dot{\gamma}}{\partial \alpha} = \frac{\rho V^2 S C_{L\alpha}}{2mV} \qquad \frac{\partial \dot{x}}{\partial \alpha} = 0$$

$$\frac{\partial \dot{\gamma}}{\partial V} = \frac{\rho S(C_{L0} + C_{L\alpha}\alpha)}{2m} + \frac{g \cos y_v}{V^2} \qquad \frac{\partial \dot{x}}{\partial V} = \cos(\gamma)$$

$$\frac{\partial \dot{\gamma}}{\partial \gamma} = \frac{g \sin \gamma}{V} \qquad \frac{\partial \dot{x}}{\partial \gamma} = -V \sin \gamma$$

$$\frac{\partial \dot{\gamma}}{\partial x} = 0 \qquad \frac{\partial \dot{x}}{\partial x} = 0$$

$$\frac{\partial \dot{\gamma}}{\partial h} = 0 \qquad \frac{\partial \dot{x}}{\partial h} = 0$$

$$\frac{\partial \dot{\gamma}}{\partial n_z} = 0 \qquad \frac{\partial \dot{x}}{\partial n_z} = 0$$

$$\frac{\partial \dot{\gamma}}{\partial T} = 0 \qquad \frac{\partial \dot{x}}{\partial T} = 0$$

$$\frac{\partial \dot{h}}{\partial \alpha} = 0$$

$$\frac{\partial \dot{h}}{\partial V} = \sin \gamma$$

$$\frac{\partial \dot{h}}{\partial \gamma} = V \cos \gamma$$

$$\frac{\partial \dot{h}}{\partial x} = 0$$

$$\frac{\partial \dot{h}}{\partial h} = 0$$

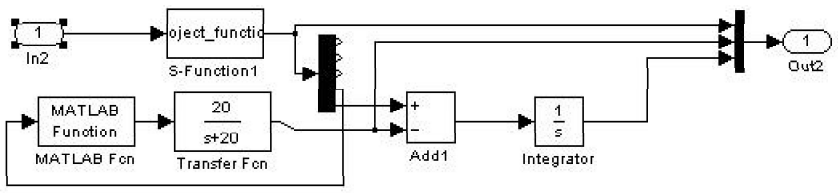
$$\frac{\partial \dot{h}}{\partial n_z} = 0$$

$$\frac{\partial \dot{h}}{\partial T} = 0$$

Once we have linearized the plant, we put in the equalibrium point. (in our case, since the operating points are so close to the equalibrium points, we are using that) Here is what the linearized plant look like.

$$\begin{pmatrix} \dot{\alpha} \\ \dot{V} \\ \dot{\gamma_v} \\ \dot{x} \\ \dot{h} \end{pmatrix} = \begin{pmatrix} -1.0153 & 0 & 0 & 0 & 0 \\ -1.8 & -.0113 & -9.8 & 0 & 0 \\ 1.0153 & .004 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 70 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \end{pmatrix} + \begin{pmatrix} .14 & 0 \\ 0 & .0001 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} n_z \\ T \end{pmatrix} + \begin{pmatrix} (\frac{\pi}{180})^2 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} w$$

In the system, we want the height of the aircraft to follow a specific trajectory. We would need to represent the error at the output of the LQR. We change our systemblocks.mdl into something like this.



Notice how the last two output are parsed out. From the states $(\alpha, V, \gamma, x, h)$, the last signal at the bottom represents the height. Our goal is for the height to remain at 400 starting from x = 10,000 till x = 15,000. Once x reaches 15,000, the aircraft starts to decend at a constant slope.

From this, the position of the aircraft determines directly the height of the aircraft. This is why we output x for the feedback into a matlab function which converts x into our expected trajectory. The matlab function looks likes this.

```
%.....................................................................
function [hei] = height(x)
if x < 15000
    hei = 400;
end
if (x > 15000)&(x<20000)
    hei = -2*x/25+1600;
end
if (x > 20000)
     hei = 0;
end
%.....................................................................
```

Once the x position is converted into the height, we can compare the current height versus the expected height for the error. We put the expected height through a shaping filter to low-pass the frequencies. Like a capacitor, if there's a spike due to noise, we can prevent it from spiking too high. We also integrate the error signal to output for the generalized plant.

Now that we have the generalized system, we need to now form the generalized plant.

We let the state of the shaping filter be called s_x , output be called s_i and input as r .

We also let the state of the erro be ε_x, output be ε_i and input as ε.

Those sections of the system are governed by these equations.

We use Kalman's formula to convert the transfer function

$$D + C(SI - A)^{-1} B$$

shaping filter

$$\frac{20}{s+20}$$

$$\dot{s}_x = -20 s_x + 20 r$$

$$s_i = s_x$$

integrator

$$\frac{1}{s}$$

$$\dot{\varepsilon}_x = 0\varepsilon_x + \varepsilon$$

$$\int \varepsilon = \varepsilon_x$$

From these equations, we can get the generalized plant.

$$\begin{pmatrix} \dot{\alpha} \\ \dot{V} \\ \dot{\gamma_v} \\ \dot{x} \\ \dot{h} \\ \dot{s_x} \\ \dot{\varepsilon_x} \end{pmatrix} = \begin{pmatrix} -1.0153 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1.8 & -.0113 & -9.8 & 0 & 0 & 0 & 0 \\ 1.0153 & .004 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 70 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -20 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \\ s_x \\ \varepsilon_x \end{pmatrix} + \begin{pmatrix} .14 & 0 & 0 \\ 0 & .0001 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 20 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} n_z \\ T \\ r \end{pmatrix} + \begin{pmatrix} (\frac{\pi}{180})^2 \\ 1 \\ n \\ n \\ n \\ n \\ n \end{pmatrix} w$$

$$\begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \\ s_i \\ \int \varepsilon \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \\ s_x \\ \varepsilon_x \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} n_z \\ T \\ r \end{pmatrix} \qquad \begin{pmatrix} n \\ n \\ n \\ n \\ n \\ n \\ n \end{pmatrix} w \qquad +$$

$$\begin{pmatrix} (\frac{\pi}{180})^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (\frac{\pi}{180})^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 70 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & n & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & n \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{pmatrix}$$

With this plant, we need to remove all the exogenous inputs for LQR.

$$\begin{pmatrix} \dot{\alpha} \\ \dot{V} \\ \dot{\gamma_v} \\ \dot{x} \\ \dot{h} \\ \dot{s_x} \\ \dot{\varepsilon_x} \end{pmatrix} = \begin{pmatrix} -1.0153 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1.8 & -.0113 & -9.8 & 0 & 0 & 0 & 0 \\ 1.0153 & .004 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 70 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -20 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \\ s_x \\ \varepsilon_x \end{pmatrix} + \begin{pmatrix} .14 & 0 \\ 0 & .0001 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} n_z \\ T \end{pmatrix}$$

$$\begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \\ s_x \\ \varepsilon \\ \int \varepsilon \\ n_z \\ T \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \\ s_x \\ \varepsilon_x \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} n_z \\ T \end{pmatrix}$$

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= C^T \text{weight}\, C$$

$$R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}^T \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$= D^T \text{weight}\, D$$

The weighting matrix for Q and R determines how important relative to other variables we need to put on that specific variable. Tbe variables we are trying to minimize consist of the output of the generalized plant for LQR.

$$\begin{pmatrix} \alpha \\ V \\ \gamma \\ x \\ h \\ s_x \\ \varepsilon \\ \int \varepsilon \\ n_z \\ T \end{pmatrix}$$

Since we have 10 things to weight, we have a weighting matrix size of 10. When we have an identity matrix, it means that every element is weighted equally. As you can see from the weighting matrix, for Q, the part where we weight x, h are set to zero. This is because we don't want to minimize them. We want the aircraft to move foward in the x direction. We want the height to remain to be 400 for a while. So we gave them a weight of zero.

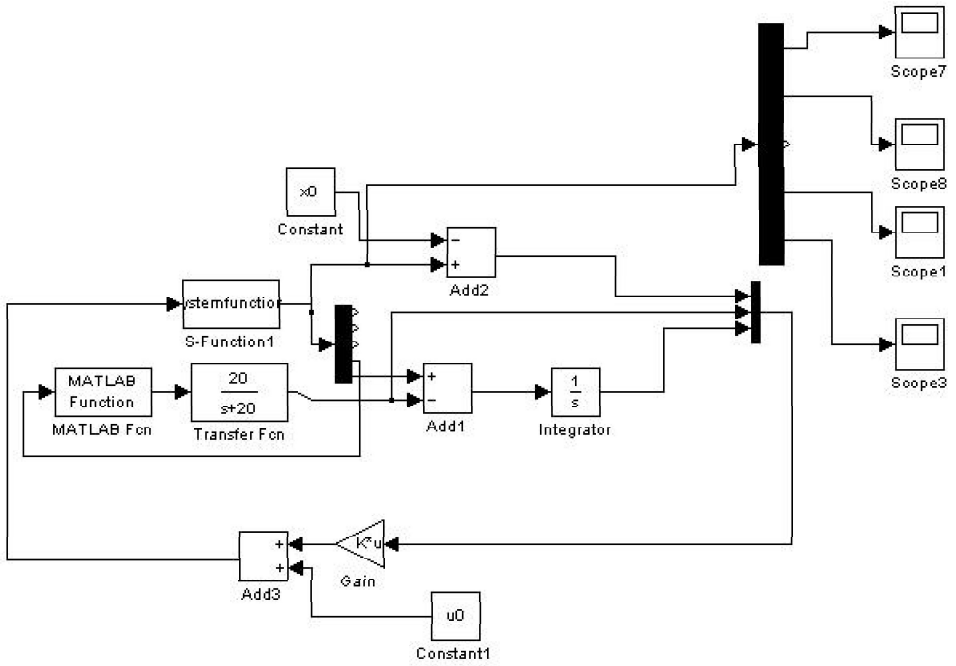
You should now go back to the systemdata.m and erase the part where you simulate the system. We only use that to check if the system is working correctly. After that, put in the A,B,C,D matrix for the generalized plant.

sys = ss(A,B,C,D);

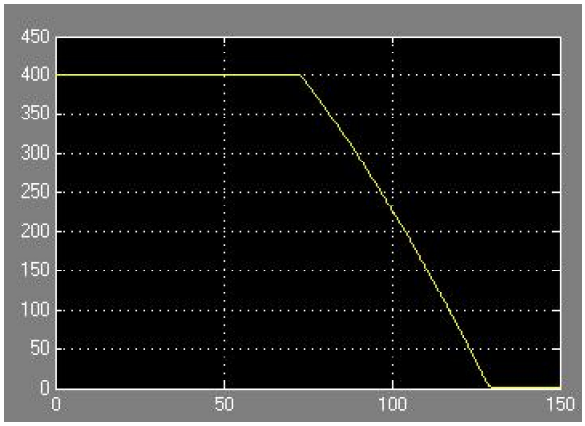
[K,S,E] = LQR(sys,Q,R);

K = -K;

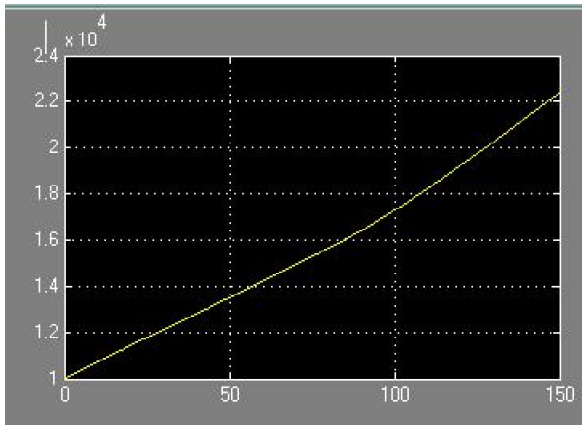
Take notice that K = - K. This is because the lqr gives the NEGATIVE. It is very important that you inver the lqr before using it. The system will blow up if you don't. I have done it many times.



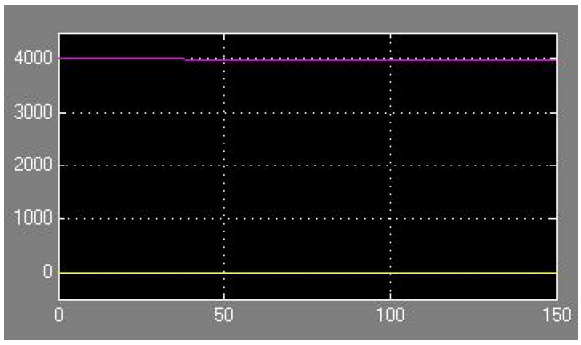
If we looked at the output at scope3 (height) this is what we will see.



If we look at scope 1, the (x trajectory) this is what we'll see.



If we have a scope for the feedback after the gain, it would look something like this. The yellow line is n_z and the pink line is the thrust. The n_z value is definitely between

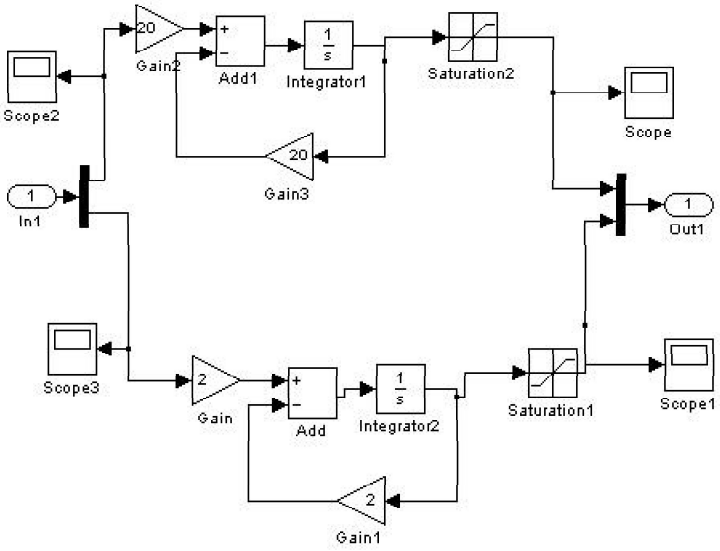


The next step is to add the measurement noise. Add a noise block at the output of the system.

where v_{ac} is white noise with covariance:

$$v_{\text{ac}} = \begin{pmatrix} (\frac{\pi}{180})^2 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & (\frac{\pi}{180})^2 & 0 & 0 \\ 0 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{pmatrix}$$

After you have added the noise, add the actuator before the system



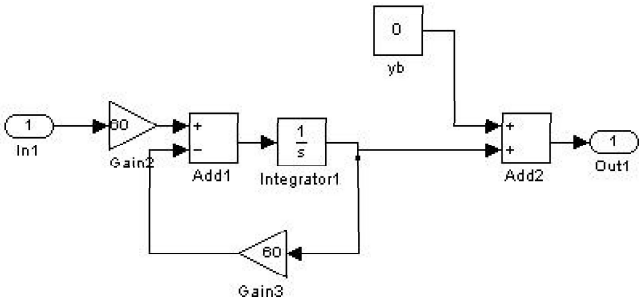
The actuator accepts commands from the controller and drive the physical control inputs. For example, in an aircraft, there are hydraulic actuators that drive wing surfaces such as elevator and rudder. Though the wing surfaces are the physical control input, we often use commands to the actuator as the control inputs for controller design. Here is the transfer function of the actuator.

$$u_p = \frac{\lambda}{s+\lambda} u_c$$

u_p is the output of the actuator which goes into the system

u_c is the input of the actuator

We also need to model the sensor dynamics which looks like this.



The equation that models the sensor dynamics is the following

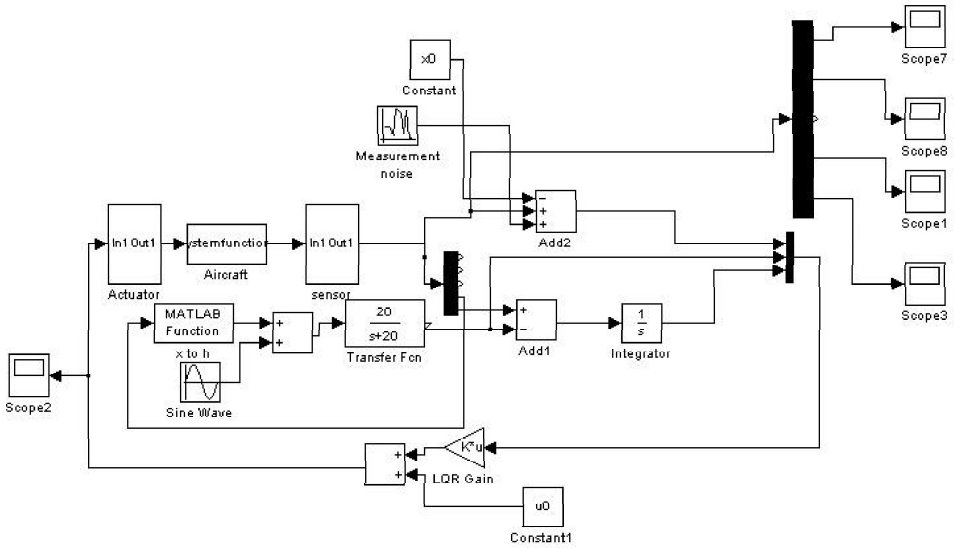
$$\dot{x}_s = -\mu x_a + \mu y_p$$

$$y_m = x_s + b + v$$

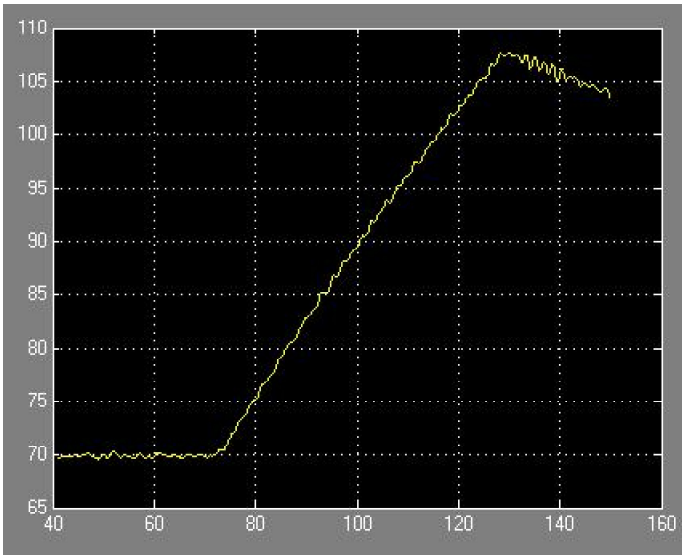
In our case, the bias y_b is equal to zero. If the bias were not zero, we would have needed to use a kalman filter to find out the bias. Since the bias was zero, a kalman filter was unnecessary.

The last aspect we need to take into consideration is the up and down motion of the carrier due to the waves. This makes the height of the carrier move up and down. This could be tracked by adding it into our compared height.

After adding all the remaining blocks into the system. The final system should look something like this.



At scope9, we have the air speed. Notice how it never went below 50. This is a good thing because you don't want the aircraft to slow down too much.



At scope2, we have the inputs which are thrust and n_z. The thrust stays close to 4000, while n_z stays relatively close to zero.